08

Implementación de Interfaces

Transcripción

[00:00] Entonces vamos a quitarle el cuerpo a este método y dejar este otro método sin cuerpo también. Y aquí hay algo curioso, si se han dado cuenta. El primer método no es abstracto y está todo bien, y el segundo método es abstracto y está todo bien. ¿Qué quiere decir esto?

[00:23] En la interfaz prácticamente todos los métodos son abstractos, esa es una característica de la interfaz, todos los métodos son abstractos, toda interfaz es abstracta. ¿Por qué? Porque no está implementada. Incluso aquí clave si se dan cuenta está como privado, y está marcándome error. ¿Por qué? Porque es un atributo privado y la interfaz no puede tener atributos privados.

[00:51] ¿Por qué? Porque no va a haber forma de accesar a ese atributo privado porque no hay cuerpo en cada método. Y al ser privado es inaccesible desde otra parte del código, entonces ahora lo borramos. Perfecto. Y ya que toda interfaz es abstracta, nosotros podemos olvidar el modificador abstract y de igual forma en los métodos.

[01:19] No tenemos el abstract porque por defecto todos los métodos son abstractos. Puedes escribirlo. ¿Lo acepta? Sí, pero no es necesario y por buena práctica no se escribe. Ahora, ¿la interfaz puede extender de funcionario? No. Una interfaz no puede extender de una clase. Una interfaz puede extender de otra interfaz, pero jamás de una clase. Entonces esto se muere.

[01:45] Y aquí ya estamos comenzando a modelar nuestra estructura. ¿Por qué? Porque estamos ya rompiendo completamente. Recuerden que autenticable

extendía de funcionario. Ya estamos rompiendo con esa relación, estamos separando los dominios y eso es bueno, eso es bueno. Vemos que saltaron errores y vamos a ver cómo lo solucionamos.

[02:10] Guardamos acá, vemos que automáticamente administrador, cliente, gerente y las pruebas colapsaron. No se preocupen, vamos primero con el cliente, cerramos lo demás y bueno, volvemos aquí arriba y nos dice: "No puedo extender de autenticable". Tenemos nuestra interfaz autenticable y nos dice: "No puedo extender de autenticable". ¿Por qué?

[02:37] Bueno. Porque autenticable no es una clase, entonces yo no puedo extender de algo que no es una clase. No puedo ser hijo de algo que no es una clase. Pero la interfaz me permite implementarla. Entonces, yo puedo cambiar extends por implements. Y si yo implemento autenticable, al igual que en la clase abstracta, yo estoy obligado a implementar los métodos de esa interfaz.

[03:08] Entonces si yo le doy a implemented methods, voy a bajar acá y ya me está dando, bueno, el getBonificacion no me servía acá porque este es un cliente, entonces ya puedo borrarlo. ¿Por qué? Porque yo ya rompí la relación de autenticable-funcionario, entonces yo puedo iniciar sesión y setearme la clave que yo quiera y eso es muy bueno. ¿Por qué?

[03:38] Porque en el caso del cliente él ya no tiene ningún tipo de relación con funcionario. Cliente ya no tiene nada que ver con el funcionario, no tiene absolutamente ya nada que ver, rompimos esa relación. Vamos ahora con el administrador. En el caso del administrador va a ser la misma historia, vamos a darle un implement. Autenticable nos va a pedir: "Oye, implementa dos métodos de la interfaz".

[04:09] Perfecto, pero ahora tenemos un plus. ¿Cuál es? Tenemos disponible nuestro extends y aquí entra la magia. Aquí entra digamos, perdón, esto era antes. Aquí entra extends Funcionario. Y aquí entra la importancia de que no puedas extender de más de una clase. Sintácticamente y conceptualmente es

correcto porque el administrador es un funcionario, entonces sí, aquí sí debe haber herencia.

[04:53] Pero la interfaz, que es autenticable, ¿qué me permite? Me permite identificar, rotular a cada funcionario o a cada objeto en este caso, lo rotulo con que: tú sí puedes entrar, tú no puedes entrar, tú sí puedes iniciar sesión, tú no puedes iniciar sesión. Solamente porque los estoy rotulando con esta interface, es como que les pongo un sello y les digo: "Tú eres autenticable".

[05:20] Fuera, no importa el tipo de objeto que seas. Le digo "Tú eres autenticable y me implementas tus métodos para que te autentiques. Entonces, independientemente de qué seas, te autenticas". Perfecto. Entonces tenemos nuestro getBonificacion, que tenemos que obviamente implementarlo, al igual que el setClave y el iniciarSesion. El mismo caso para gerente.

[05:50] Gerente va a implementar de autenticable y extender de funcionario. ¿Por qué? Porque el gerente es un funcionario y es autenticable. Para que compile agregamos los métodos y listo, todo bien. Perfecto. Y vemos claramente que aquí se fueron todos los errores. Y vámonos a nuestro Test de sistema interno, que aquí hay algo curioso que yo quiero que vean.

[06:22] En el sistema interno, si se dan cuenta, cuando mandamos el método autentica, nosotros mandamos al gerente y al administrador. Vamos a entrar al método otra vez. Vemos que él recibe autenticable como parámetro, pero autenticable es una interfaz, no es un objeto. Entonces ese es otro de los beneficios de la interfaz. Puedes usar ese rótulo.

[06:50] Recuerda que la interfaz es un rótulo que le pones al objeto y decirle: "Todo lo que tenga este sello, todo lo que tenga este rótulo, esta etiqueta, va a ejecutar este método". Todo lo que sea autenticable. Indiferentemente si eres gerente, si eres funcionario, si eres cliente, si eres proveedor, todos esos que estén rotulados como autenticable, van a ejecutar este método.

[07:15] Y tiene mi método iniciarSesion. Pero ojo, este iniciarSesion, si bien está indicado aquí, él no sabe cómo va a ser efectuado, porque autenticable te indica la firma del método pero no te dice cómo lo vas a ejecutar. ¿Eso depende de quién? De cada uno. Depende del administrador, depende del gerente y va a depender del cliente, entonces este método que yo tenía aquí en mente, ya es completamente posible.

[07:48] Esta estructura es completamente posible, es escalable, porque vemos claramente que el concepto, o perdón, la cohesión que había entre esos conceptos, ya está rota, ya no hay. Y recuerden que uno de los pilares de la programación orientada a objetos es alta cohesión, bajo acoplamiento. Y el acoplamiento que había entre autenticable y funcionario se fue. Ya no existe. Ahora son dos conceptos distintos y el sistema ya puede escalar ya exponencialmente.